UCLA Computer Science

STAR AI RESEARCH LAB UCLA

Reasoning about Learned Models' Behavior

Guy Van den Broeck

ODSC - Nov 18, 2021

The AI Dilemma

Pure (Logic) Reasoning ⟷ **Pure Learning**

The AI Dilemma

Pure (Logic) Reasoning **Pure Learning**

- Slow thinking: deliberative, cognitive, model-based, extrapolation
- Amazing achievements until this day

- "*Pure logic is brittle*"

  noise, uncertainty, incomplete knowledge, …

The AI Dilemma

Pure (Logic) Reasoning **Pure Learning**

- Fast thinking: instinctive, perceptive, model-free, interpolation
- Amazing achievements recently
- "*Pure learning is brittle*"

bias, algorithmic fairness, interpretability, explainability, adversarial attacks, unknown unknowns, calibration, verification, missing features, missing labels, data efficiency, shift in distribution, general robustness and safety

fails to incorporate a sensible model of the world

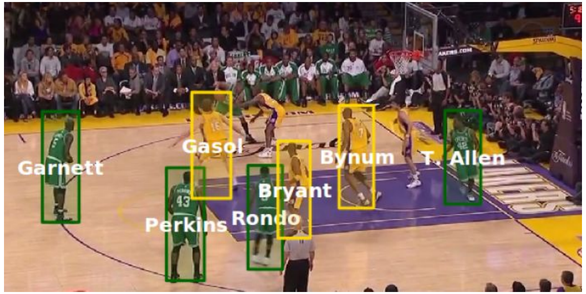
The AI Dilemma

- Learn statistical models subject to symbolic knowledge
- Integrate reasoning into modern learning algorithms

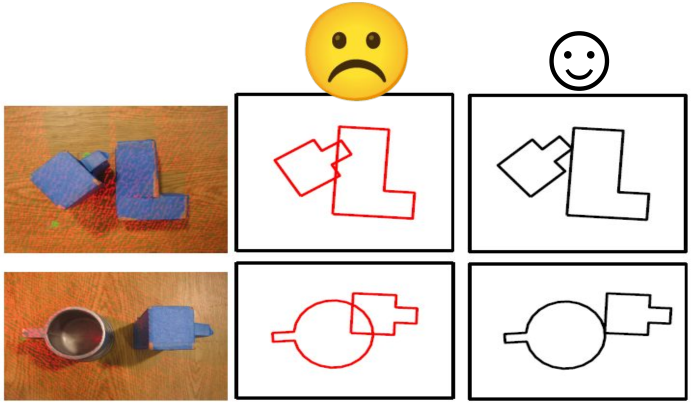
Today: Deep learning with constraints
Learning monotonic neural networks

Deep Learning with Constraints

Knowledge in Vision, Robotics, NLP



People appear at most once in a frame

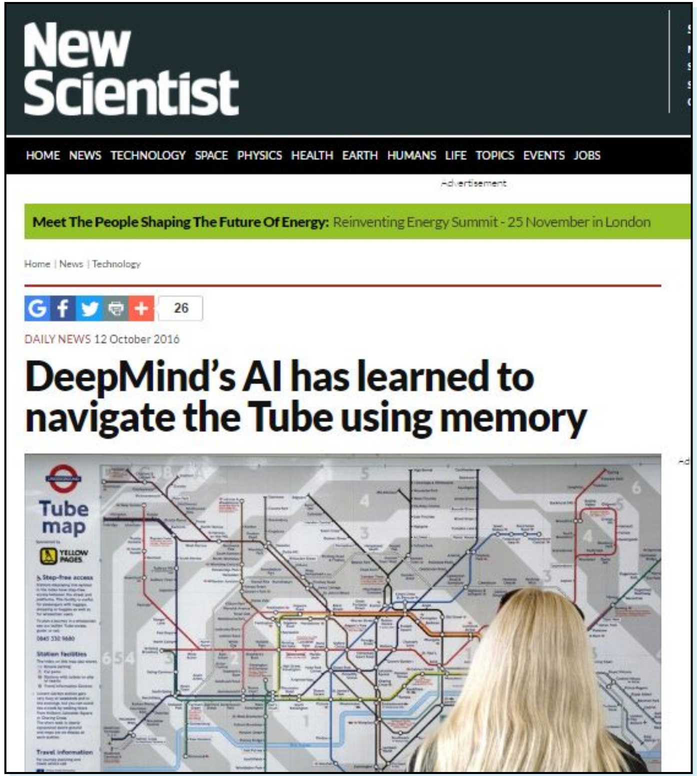


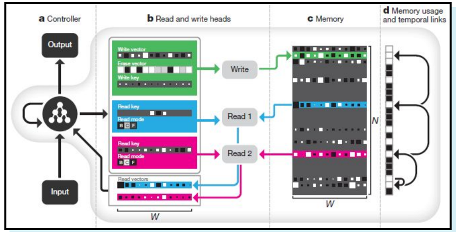
Rigid objects don’t overlap

At least one verb in each sentence.
If X and Y are married, then they are people.

[Lu, W. L., Ting, J. A., Little, J. J., & Murphy, K. P. (2013). Learning to track and identify players from broadcast sports videos.], [Wong, L. L., Kaelbling, L. P., & Lozano-Perez, T., Collision-free state estimation. ICRA 2012], [Chang, M., Ratinov, L., & Roth, D. (2008). Constraints as prior knowledge], [Ganchev, K., Gillenwater, J., & Taskar, B. (2010). Posterior regularization for structured latent variable models]… and many many more!

Motivation: Deep Learning



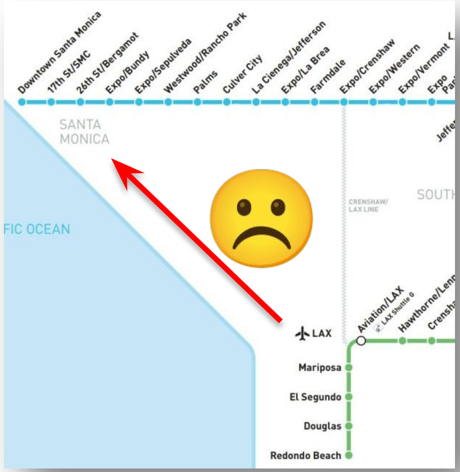


[Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., et al.. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471-476.]

Motivation: Deep Learning

DeepMind's latest technique uses external memory to **solve tasks that require logic and reasoning — a step toward more human-like AI.**

… but …



optimal planner recalculating a shortest path to the end node. To ensure that the network always moved to a valid node, the output distribution was renormalized over the set of possible triples outgoing from the current node. The performance

it also received input triples during the answer phase, indicating the actions chosen on the previous time-step. This makes the problem a 'structured prediction'

[Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., et al.. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471-476.]

Knowledge vs. Data

- Where did the world knowledge go?
  - Python scripts
    - Decode/encode cleverly
    - Fix inconsistent beliefs
  - Rule-based decision systems
  - Dataset design
  - “a big hack” (with author’s permission)

Knowledge vs. Data

- Where did the world knowledge go?
  - Python scripts
    - Decode/encode cleverly
    - Fix inconsistent beliefs
  - Rule-based decision systems
  - Dataset design
  - “a big hack” (with author’s permission)
- In some sense we went backwards

  Less principled, scientific, and intellectually satisfying ways of incorporating knowledge

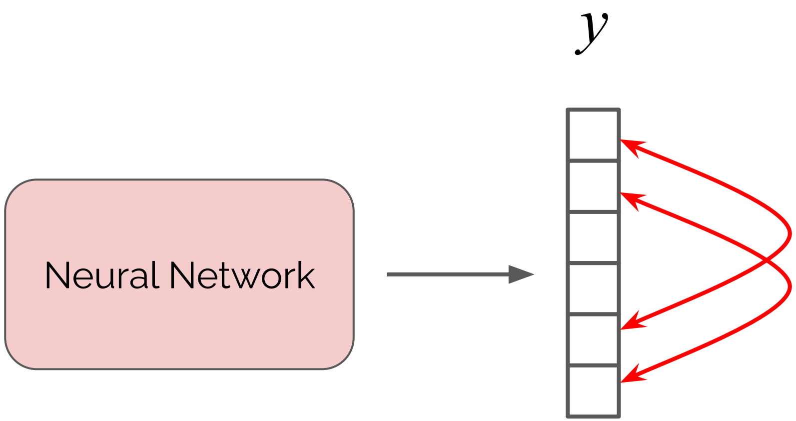
pylon

A PyTorch Framework for Learning with Constraints

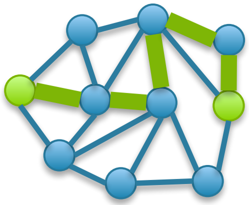
Kareem Ahmed Tao Li Thy Ton Quan Guo,
Kai-Wei Chang Parisa Kordjamshidi Vivek Srikumar
Guy Van den Broeck Sameer Singh

http://pylon-lib.github.io

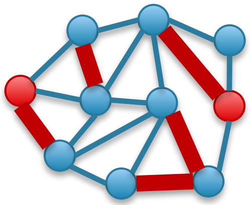
Declarative Knowledge of the Output



How is the output structured?
Are all possible outputs valid?



vs.



How are the outputs related to each other?

Learning this from data is inefficient
Much easier to express this declaratively

How can do we inject declarative knowledge into PyTorch training code?

pylon

Library that extends PyTorch to allow injection of declarative knowledge

- **Easy to Express Knowledge:** users write arbitrary constraints on the output
- **Integrates with PyTorch:** minimal change to existing code
- **Efficient Training:** compiles into loss that can be efficiently optimized
    - Exact semantic loss (see later)
    - Monte-carlo estimate of loss
    - T-norm approximation
    - *your solver?*

pylon

PyTorch Code

```
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)
```

1 Specify knowledge as a predicate

```
def check(y):
    ...
    return isValid
```

pylon

1 Specify knowledge as a predicate

```
def check(y):
    ...
    return isValid
```

PyTorch Code

```
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)

    loss += constraint_loss(check)(py)
```

2 Add as loss to training

```
loss += constraint_loss(check)
```

pylon

PyTorch Code

```
for i in range(train_iters):
    ...
    py = model(x)
    ...
    loss = CrossEntropy(py,...)

    loss += constraint_loss(check)(py)
```

1 Specify knowledge as a predicate

```
def check(y):
    ...
    return isValid
```

2 Add as loss to training

```
loss += constraint_loss(check)
```

3 pylon derives the gradients
(solves a combinatorial problem)

http://pylon-lib.github.io

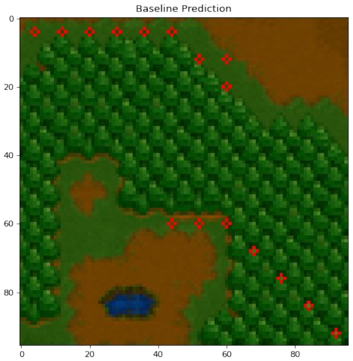
Warcraft Shortest Path

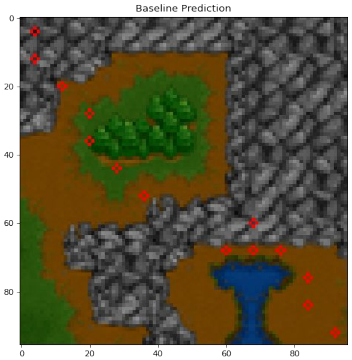
Predicting the min-cost simple-path in a grid

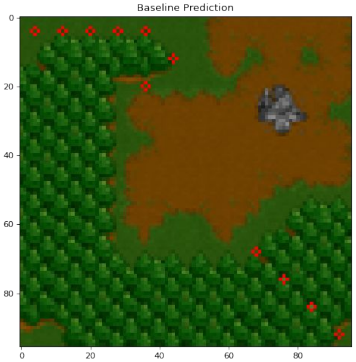
[Vlastelica et al., 2019]

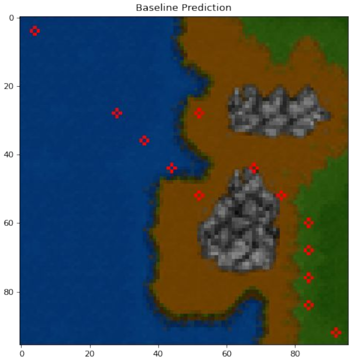
without constraint

without constraint

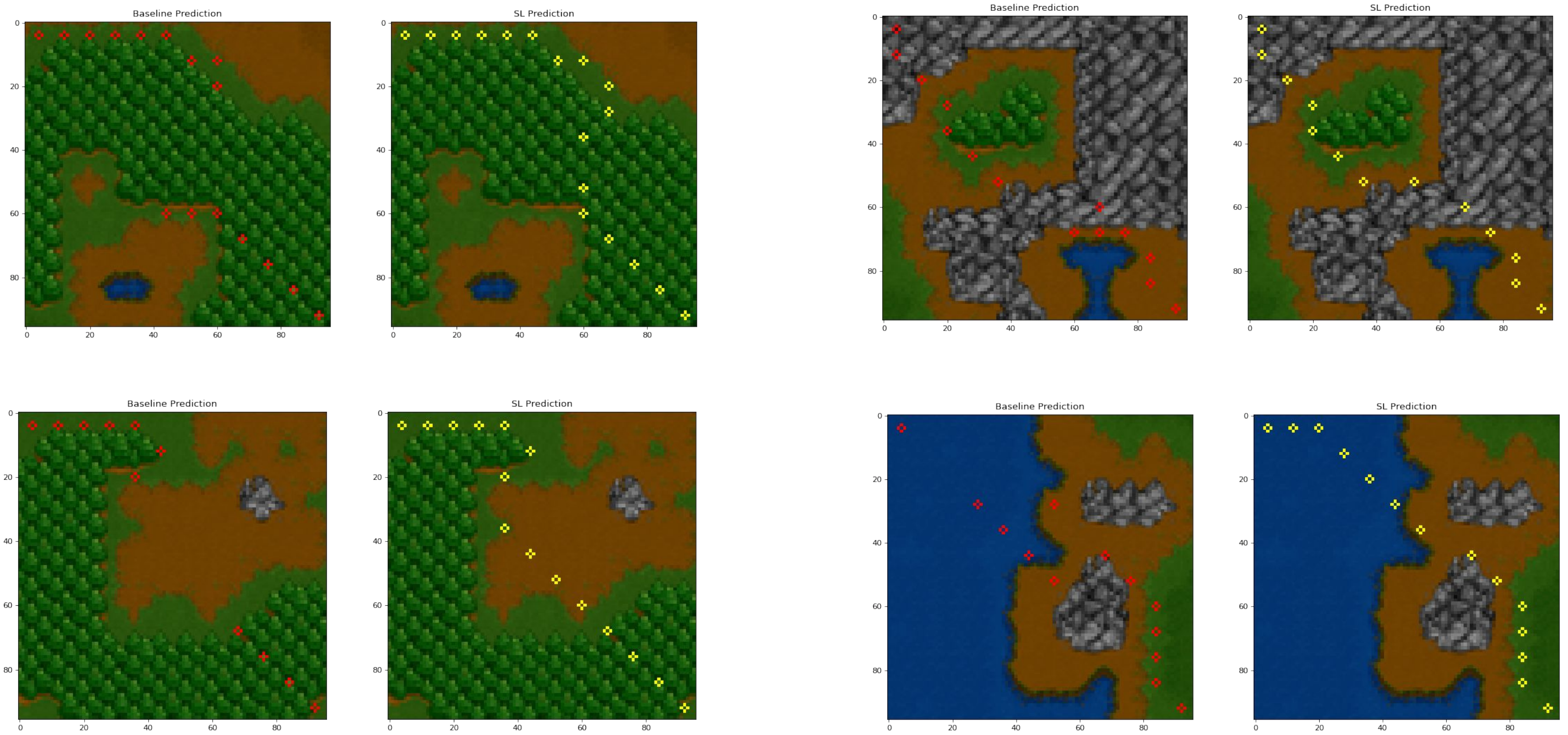








without constraint *with constraint* *without constraint* *with constraint*



Warcraft min-cost simple-path prediction results

Test accuracy %	Coherent	Incoherent	Constraint
ResNet-18	44.8	**97.7**	56.9

Is prediction the shortest path?
This is the real task!

Are individual edge predictions correct?

Is output a path?

Warcraft min-cost simple-path prediction results

Test accuracy %	Coherent	Incoherent	Constraint
ResNet-18	44.8	**97.7**	56.9
+ Semantic loss	**50.9**	**97.7**	**67.4**

Semantic Loss

<u>Q</u>: How close is output ***p*** *to satisfying constraint α?*

*<u>A</u>: Semantic loss function L(α,**p**)*

- Axioms, for example:
  - If α constrains to one label, L(α,**p**) is cross-entropy
  - If α implies β then L(α,**p**) ≥ L(β,**p**)   (*α more strict*)

- Implied Properties:
  - If α is equivalent to β then L(α,**p**) = L(β,**p**)
  - If **p** is Boolean and satisfies α then L(α,**p**) = 0

SEMANTIC
Loss!

Axioms imply unique semantic loss:

$$\underbrace{\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \propto -\log \sum_{\mathbf{x} \models \alpha} \prod_{i:\mathbf{x} \models X_i} \mathsf{p}_i \prod_{i:\mathbf{x} \models \neg X_i} (1 - \mathsf{p}_i)}$$

Probability of satisfying constraint α after sampling from neural net output layer **p**

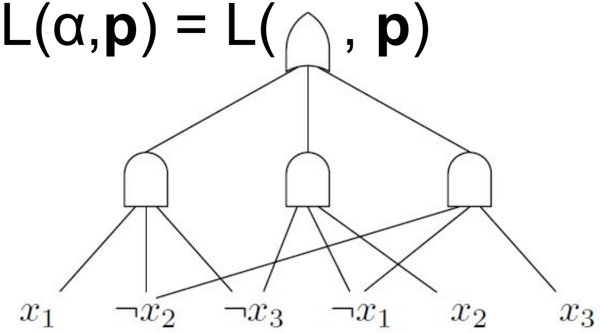
In general: #P-hard ☹️

We do this probabilistic-logical reasoning during learning in a computation graph

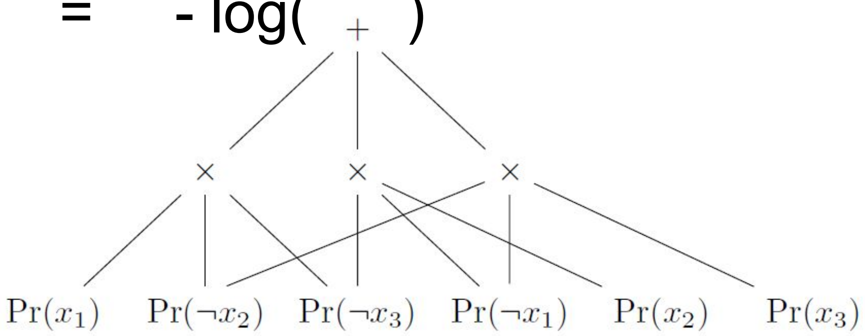
Logical Computation Graphs

- Logical circuits that can count solutions (#SAT)
- Also compute semantic loss efficiently in size of circuit

$L(\alpha,\mathbf{p}) = L($

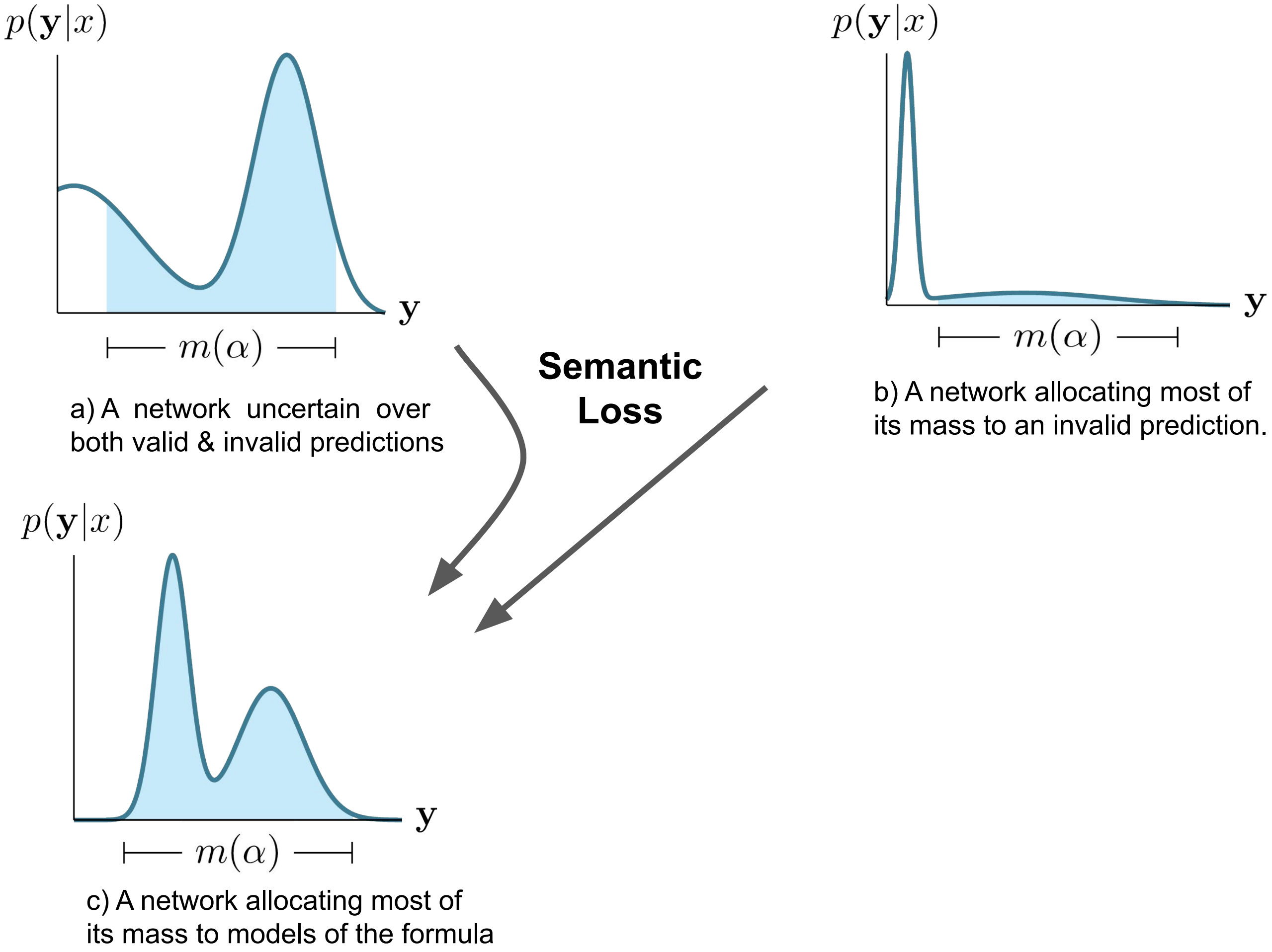


$, \mathbf{p}) = -\log($

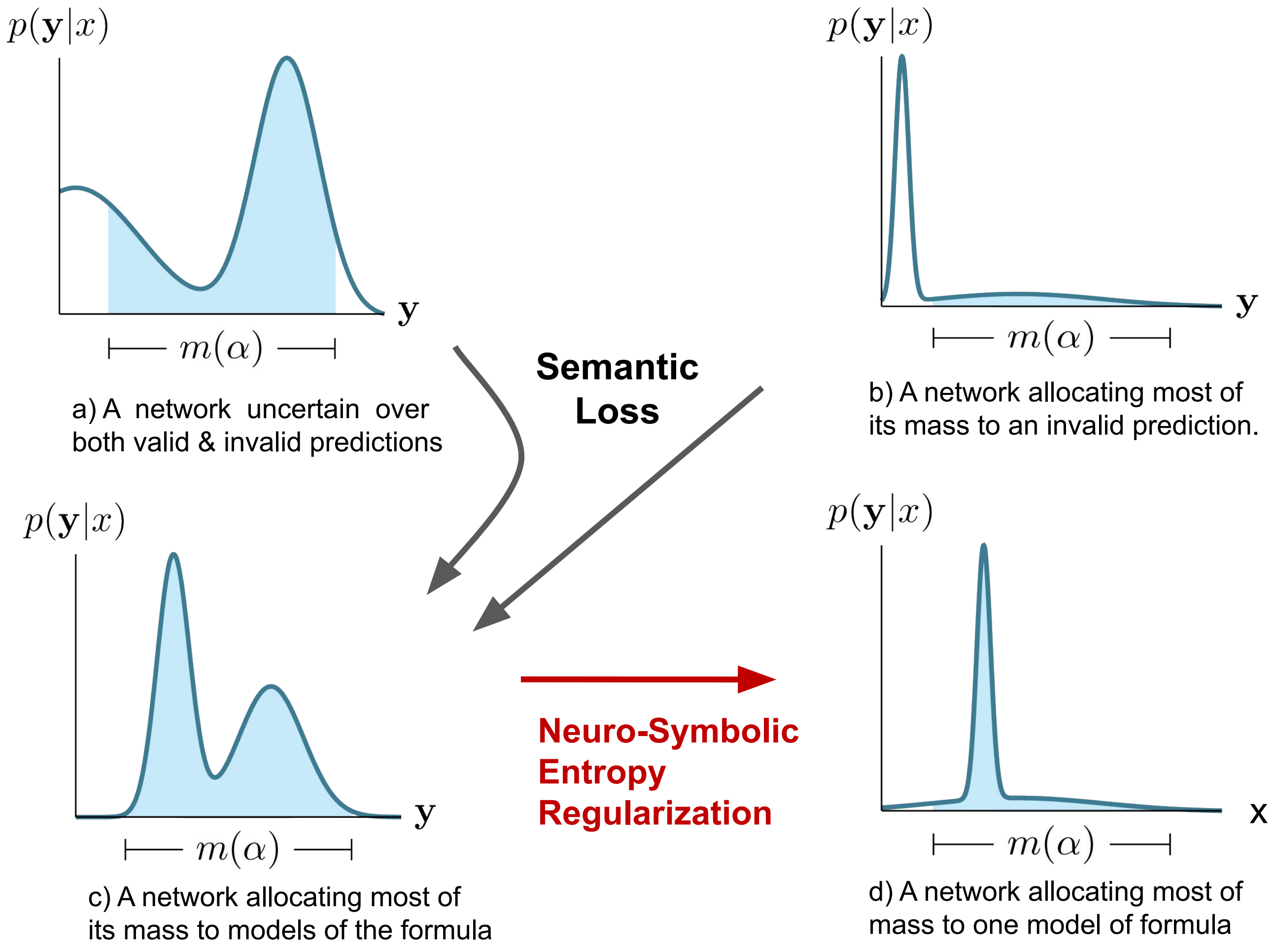


$)$

- Compilation into circuit by SAT solvers (once)
- Add circuit to neural network output in pytorch/tensorflow/...



$p(\mathbf{y}|x)$
$\mathbf{y}$
$m(\alpha)$
a) A network uncertain over both valid & invalid predictions
Semantic Loss
$p(\mathbf{y}|x)$
$\mathbf{y}$
$m(\alpha)$
b) A network allocating most of its mass to an invalid prediction.
$p(\mathbf{y}|x)$
$\mathbf{y}$
$m(\alpha)$
c) A network allocating most of its mass to models of the formula



p(y|x)
y
m(α)
a) A network uncertain over both valid & invalid predictions
p(y|x)
y
m(α)
b) A network allocating most of its mass to an invalid prediction.
Semantic Loss
p(y|x)
y
m(α)
c) A network allocating most of its mass to models of the formula
Neuro-Symbolic Entropy Regularization
p(y|x)
x
m(α)
d) A network allocating most of mass to one model of formula

Two complementary neuro-symbolic losses

Neural net $P(\mathbf{Y}|x)$

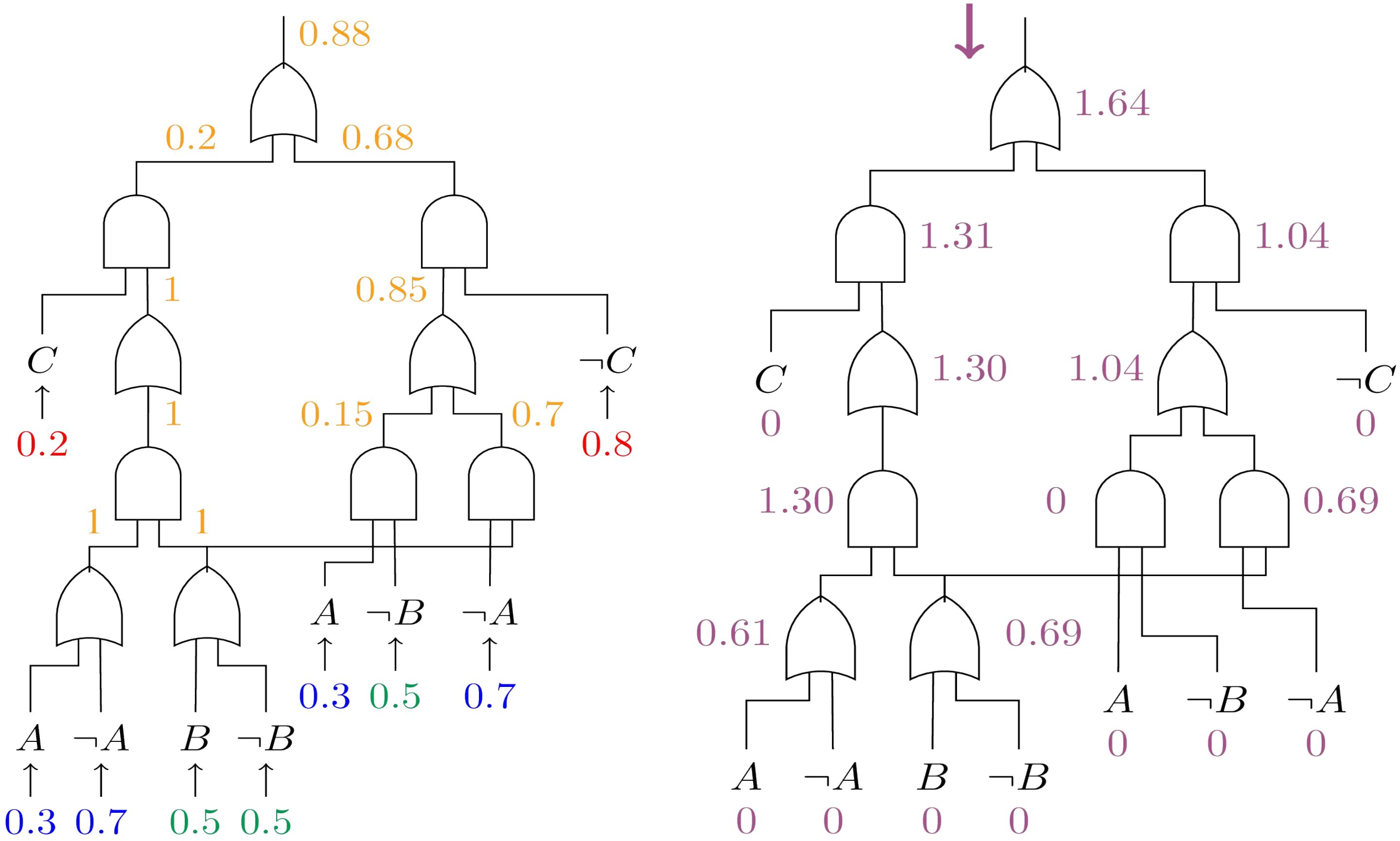
Constraint α

$$P(\mathbf{Y}|x), \alpha \xrightarrow{\textbf{Semantic Loss}} P(\alpha|x)\uparrow: -\log P(\alpha|x)\downarrow$$

$$P(\mathbf{Y}|x), \alpha \xrightarrow[\textbf{Entropy-regularization}]{\textbf{Neuro-symbolic}} -\mathbb{E}_{P(\mathbf{Y}|x,\alpha)}[\log P(\mathbf{Y}|x,\alpha)]$$

Warcraft min-cost simple-path prediction results

Test accuracy %	Coherent	Incoherent	Constraint
ResNet-18	44.8	**97.7**	56.9
Semantic loss	**50.9**	**97.7**	**67.4**
+ Entropy All	51.5	97.6	67.7
+ Entropy Circuit	**55.0**	**97.9**	**69.8**



0.88
0.2
0.68
1
0.85
C
¬C
1
0.15
0.7
0.2
0.8
1
1
A
¬B
¬A
0.3
0.5
0.7
A
¬A
B
¬B
0.3
0.7
0.5
0.5
1.64
1.31
1.04
1.30
1.04
C
¬C
0
0
1.30
0
0.69
0.61
0.69
A
¬B
¬A
0
0
0
A
¬A
B
¬B
0
0
0
0

pylon

- Joint entity-relation extraction in natural language processing
- Semantic role labeling in natural language processing
- Training MNIST recognition network from arithmetic supervision
- Training neural net to solve Sudoku
- Learning to rank
- etc.

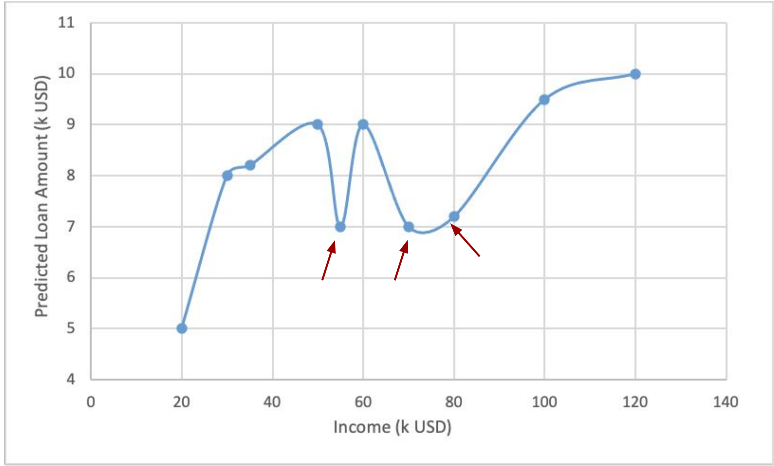
Joint entity-relation extraction in natural language processing

# Labels		3	5	10	15	25	50	75
ACE05	Baseline	4.92 ± 1.12	7.24 ± 1.75	13.66 ± 0.18	15.07 ± 1.79	21.65 ± 3.41	28.96 ± 0.98	33.02 ± 1.17
	Self-training	7.72 ± 1.21	12.83 ± 2.97	16.22 ± 3.08	17.55 ± 1.41	27.00 ± 3.66	32.90 ± 1.71	37.15 ± 1.42
	Product t-norm	8.89 ± 5.09	14.52 ± 2.13	19.22 ± 5.81	21.80 ± 7.67	30.15 ± 1.01	34.12 ± 2.75	37.35 ± 2.53
	Semantic Loss	12.00 ± 3.81	14.92 ± 3.14	22.23 ± 3.64	27.35 ± 3.10	30.78 ± 0.68	36.76 ± 1.40	38.49 ± 1.74
	+ Entropy All	$\mathbf{14.80 \pm 3.70}$	15.78 ± 1.90	23.34 ± 4.07	28.09 ± 1.46	31.13 ± 2.26	36.05 ± 1.00	39.39 ± 1.21
	+ Entropy Circuit	14.72 ± 1.57	$\mathbf{18.38 \pm 2.50}$	$\mathbf{26.41 \pm 0.49}$	$\mathbf{31.17 \pm 1.68}$	$\mathbf{35.85 \pm 0.75}$	$\mathbf{37.62 \pm 2.17}$	$\mathbf{41.28 \pm 0.46}$
SciERC	Baseline	2.71 ± 1.1	2.94 ± 1.0	3.49 ± 1.8	3.56 ± 1.1	8.83 ± 1.0	12.32 ± 3.0	12.49 ± 2.6
	Self-training	3.56 ± 1.4	3.04 ± 0.9	4.14 ± 2.6	3.73 ± 1.1	9.44 ± 3.8	14.82 ± 1.2	13.79 ± 3.9
	Product t-norm	$\mathbf{6.50 \pm 2.0}$	8.86 ± 1.2	10.92 ± 1.6	13.38 ± 0.7	13.83 ± 2.9	19.20 ± 1.7	19.54 ± 1.7
	Semantic Loss	6.47 ± 1.02	$\mathbf{9.31 \pm 0.76}$	11.50 ± 1.53	12.97 ± 2.86	14.07 ± 2.33	20.47 ± 2.50	23.72 ± 0.38
	+ Entropy All	6.26 ± 1.21	8.49 ± 0.85	11.12 ± 1.22	14.10 ± 2.79	17.25 ± 2.75	$\mathbf{22.42 \pm 0.43}$	24.37 ± 1.62
	+ Entropy Circuit	6.19 ± 2.40	8.11 ± 3.66	$\mathbf{13.17 \pm 1.08}$	$\mathbf{15.47 \pm 2.19}$	$\mathbf{17.45 \pm 1.52}$	22.14 ± 1.46	$\mathbf{25.11 \pm 1.03}$

Table 5: Experimental results for joint entity-relation extraction on ACE05 and SciERC. #Labels indicates the number of labeled data points made available to the network per relation. The remaining training set is stripped of labels and is utilized in an unsupervised manner: enforce the constraint or minimize the entropy. We report averages and errors across 3 different runs.

Monotonicity Invariants for Neural Networks

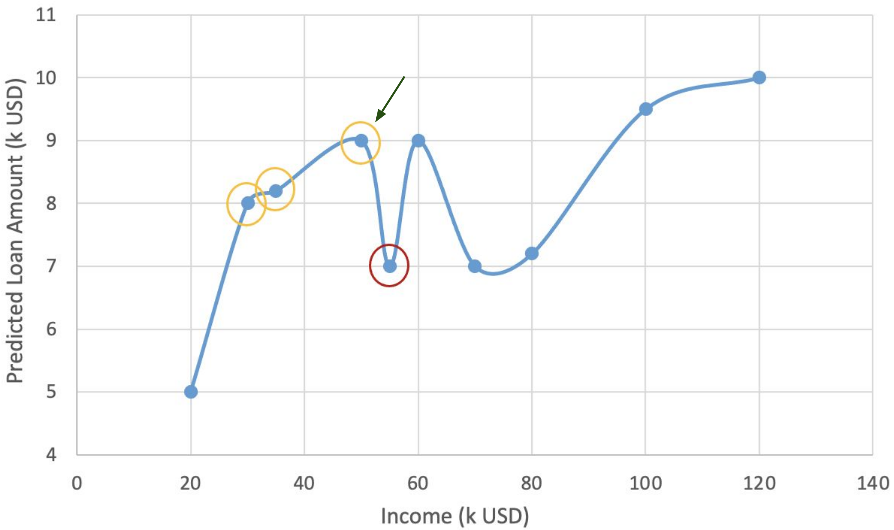
Predict Loan Amount



Neural Network Model: Increasing income can decrease the approved loan amount

Monotonicity (Prior Knowledge):
Increasing income should increase the approved loan amount

Counterexamples

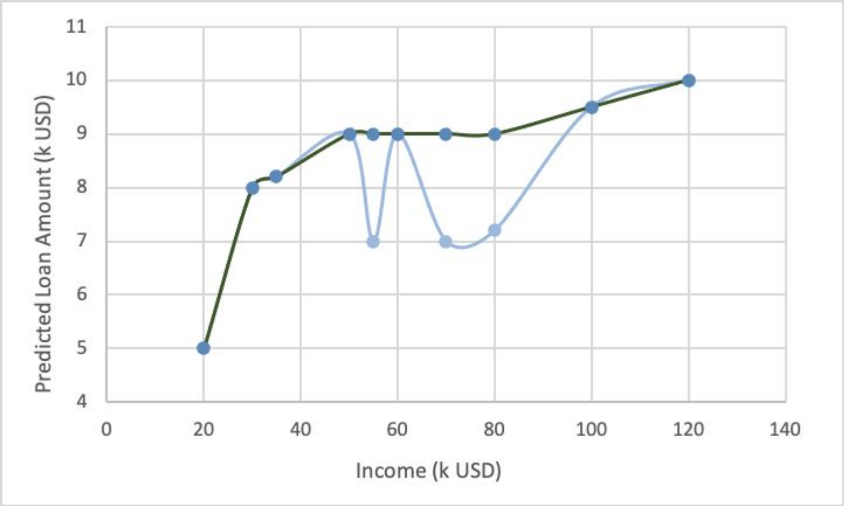


$$\exists x, y\ x \leq y \implies f(x) > f(y)$$

Computed using SMT(LRA) logical reasoning solver

Maximal counterexamples (largest violation) using OMT

Counterexample-Guided Predictions



Monotonic Envelope:

- Replace each prediction by its maximal counterexample
- Envelope construction is online (during prediction)
- Guarantees monotonic predictions for any ReLU neural net

- Works for high-dimensional input
- Works for multiple monotonic features

Monotonic Envelope: Performance

Dataset	Feature	NN_b	Envelope
Auto-MPG	Weight	9.33±3.22	**9.19±3.41**
	Displ.	9.33±3.22	9.63±2.61
	W,D	9.33±3.22	9.63±2.61
	W,D,HP	9.33±3.22	9.63±2.61
Boston	Rooms	14.37±2.4	**14.19±2.28**
	Crime	14.37±2.4	**14.02±2.17**

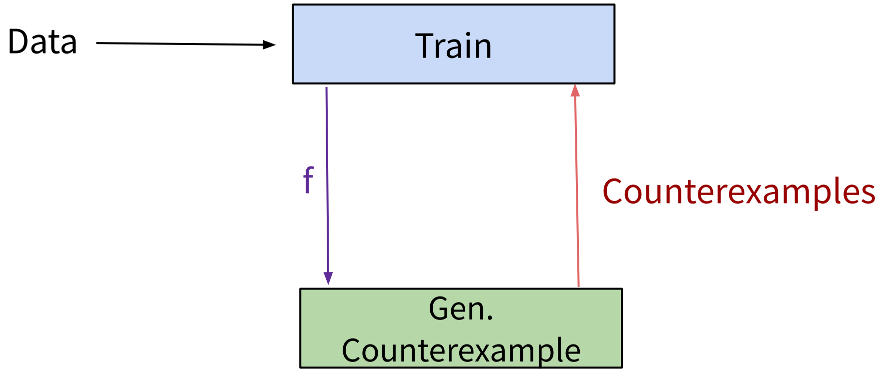
Dataset	Feature	NN_b	Envelope
Heart	Trestbps	0.85±0.04	0.85±0.04
	Chol.	0.85±0.04	0.85±0.05
	T,C	0.85±0.04	0.85±0.05
Adult	Cap. Gain	0.84	0.84
	Hours	0.84	0.84

Guaranteed monotonicity at little to no cost

Counterexample-Guided Learning

How to use monotonicity to improve model quality?

“Monotonicity as inductive bias”



T Epochs

Counterexample-Guided Learning: Performance

Dataset	Feature	NN_b	CGL
Auto-MPG	Weight	9.33±3.22	**9.04±2.76**
	Displ.	9.33±3.22	**9.08±2.87**
	W,D	9.33±3.22	**8.86±2.67**
	W,D,HP	9.33±3.22	**8.63±2.21**
Boston	Rooms	14.37±2.4	**12.24±2.87**
	Crime	14.37±2.4	**11.66±2.89**

Dataset	Feature	NN_b	CGL
Heart	Trestbps	0.85±0.04	**0.86±0.02**
	Chol.	**0.85±0.04**	**0.85±0.05**
	T,C	0.85±0.04	**0.86±0.06**
Adult	Cap. Gain	**0.84**	**0.84**
	Hours	**0.84**	**0.84**

Monotonicity is a *great* inductive bias for learning

Counterexample-Guided Monotonicity Enforced Training (COMET)

Table 4: Monotonicity is an effective inductive bias. COMET outperforms Min-Max networks on all datasets. COMET outperforms DLN in regression datasets and achieves similar results in classification datasets.

Dataset	Features	Min-Max	DLN	COMET
Auto-MPG	Weight	9.91±1.20	16.77±2.57	**8.92±2.93**
	Displ.	11.78±2.20	16.67±2.25	**9.11±2.25**
	W,D	11.60±0.54	16.56±2.27	**8.89±2.29**
	W,D,HP	10.14±1.54	13.34±2.42	**8.81±1.81**
Boston	Rooms	30.88±13.78	15.93±1.40	**11.54±2.55**
	Crime	25.89±2.47	12.06±1.44	**11.07±2.99**

Dataset	Features	Min-Max	DLN	COMET
Heart	Trestbps	0.75±0.04	0.85±0.02	**0.86±0.03**
	Chol.	0.75±0.04	0.85±0.04	**0.87±0.03**
	T,C	0.75±0.04	**0.86±0.02**	**0.86±0.03**
Adult	Cap. Gain	0.77	**0.84**	**0.84**
	Hours	0.73	**0.85**	0.84

COMET = Provable Guarantees + SotA Results

Reasoning about the Feature Distribution

Reasoning about World Model + Classifier

- “*Pure learning is brittle*”

  bias, **algorithmic fairness**, interpretability, **explainability**, adversarial attacks, unknown unknowns, calibration, verification, **missing features**, missing labels, data efficiency, shift in distribution, general robustness and safety

  fails to incorporate a sensible model of the world

- Given a learned predictor F(x) over features x
- Given a probabilistic world model P(x) - a feature distribution
- How does the world act on learned predictors?

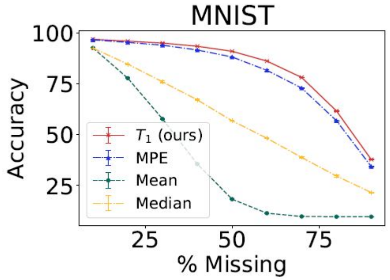
  Can we solve these hard problems?

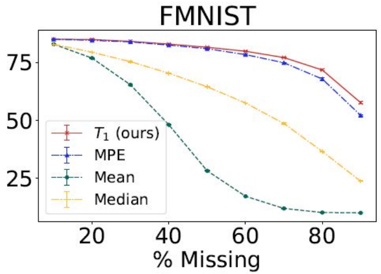
What to expect of classifiers?

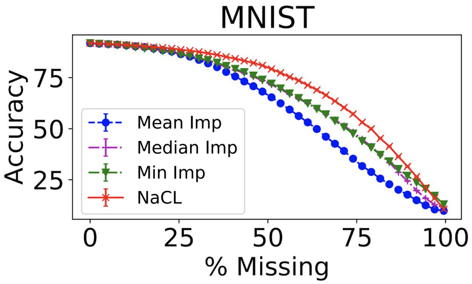
- Missing features at prediction time
- What is expected prediction of F(x) in P(x)?

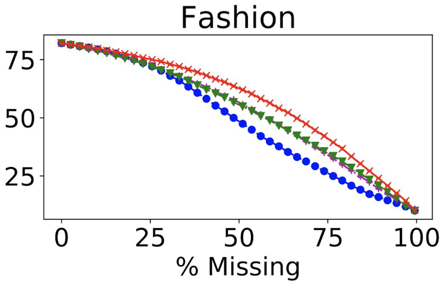
$$E_{\mathcal{F},P}(\mathbf{y}) = \mathop{\mathbb{E}}_{\mathbf{m}\sim P(\mathbf{M}|\mathbf{y})}\left[\mathcal{F}(\mathbf{ym})\right]$$

M: Missing features
y: Observed Features









Explaining classifiers on the world

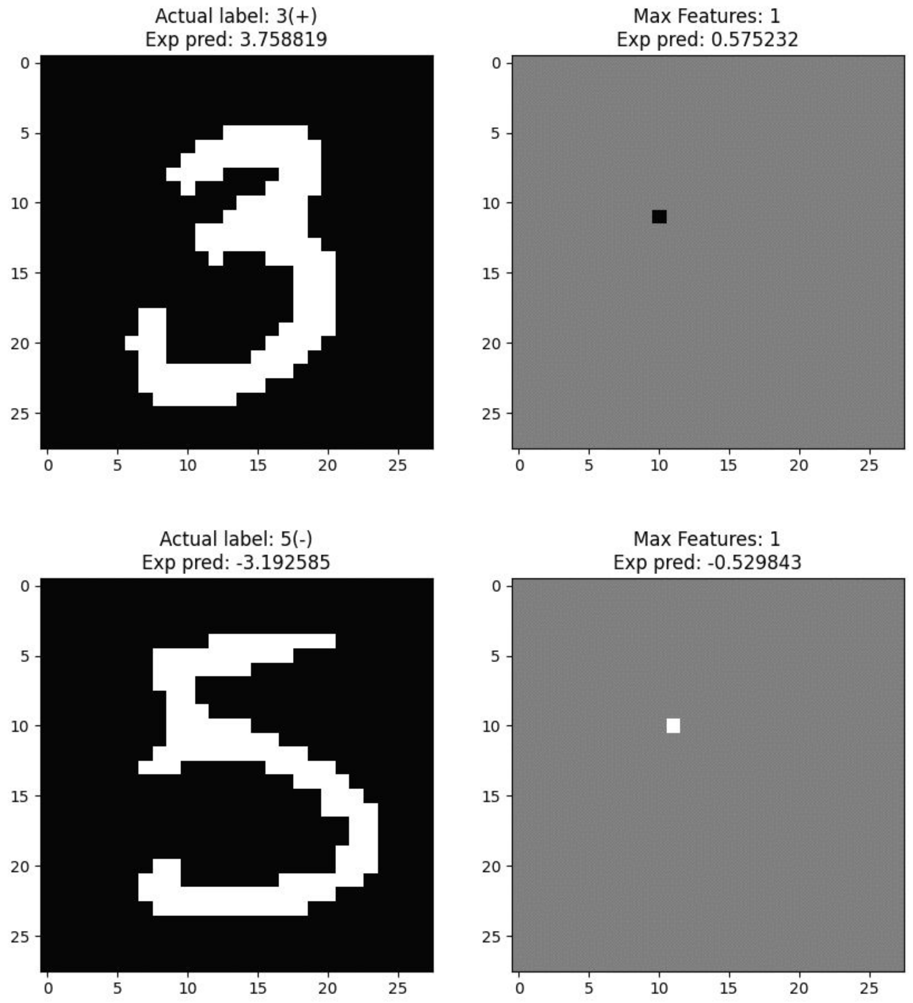
If the world looks like P(x),
*then what part of the data is **sufficient** for*
F(x) to make the prediction it makes?

Probabilistic Sufficient Explanations

Correctly Classified Examples

Binary classification: 3 vs 5

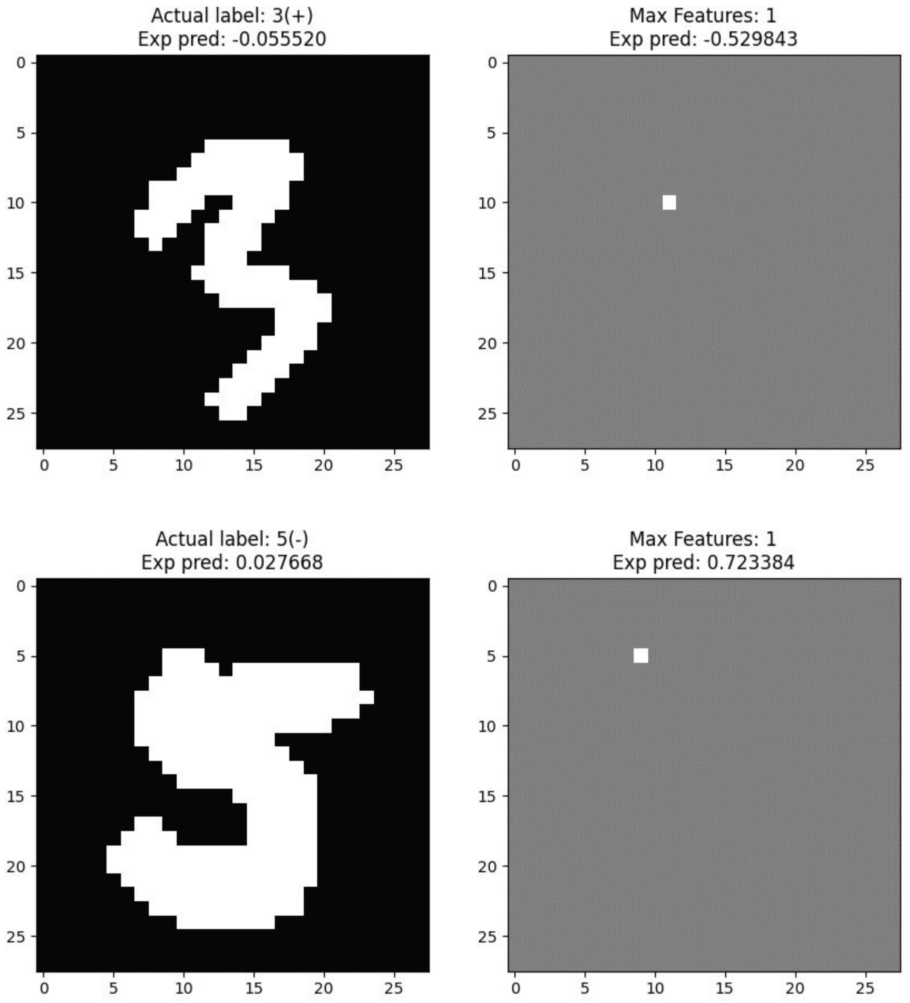
Used decision forest classifier and probabilistic circuit feature distribution



Misclassified Examples

Binary classification: 3 vs 5

Used decision forest classifier and probabilistic circuit feature distribution



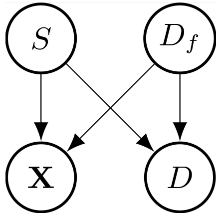
Algorithmic Fairness: Latent Fair Decisions

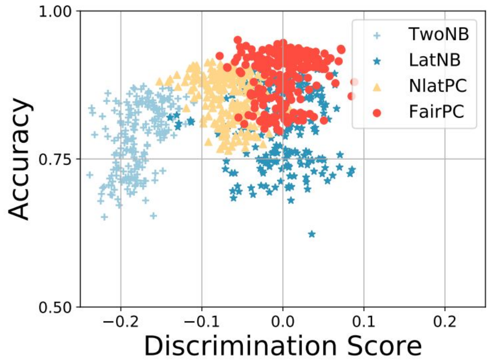
Learn classifier given

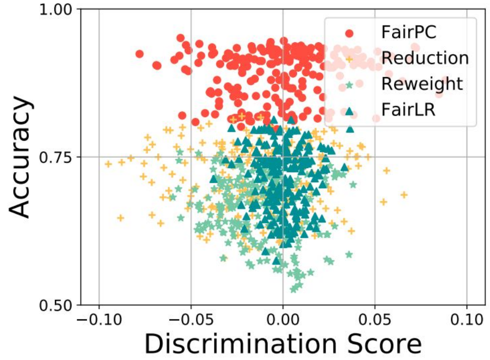
- features S and X
- training labels/decisions D

Unknown fair decision D_f should be independent of the sensitive attribute S

Discover the **latent fair decision** D_f by learning distribution P(S,X,D_f,D) where D_f is 'fair'.







competitive ***classification accuracy***

and ***better fairness guarantee***

[Choi et al. AAAI21]

The AI Dilemma

Pure (Logic) Reasoning **Pure Learning**

- Knowledge is (hidden) everywhere in ML
- A little bit of reasoning goes a long way!

Today: Deep learning with constraints
Learning monotonic neural networks

Thanks

This was the work of many wonderful students/postdoc/collaborators!

References: http://starai.cs.ucla.edu/publications/